IMPORTANCE OF NEURAL NETWORKS IN ARTIFICIAL INTELLIGENCE

Anishka Agarwal
Solid state physics laboratory (SSPL) –
Defense research and development organization (DRDO)

ABSTRACT

The role of AI is to teach the machines to learn from their mistakes and do the tasks more effectively. One of its breakthroughs is the artificial neural network (ANN) in artificial intelligence. An Artificial Neural Network (ANN) is a computational model inspired by the way biological neural networks or we can say neurons in the human brain process information. Neural network is a sub-domain of Machine Learning. It is specifically designed to recognize patterns and relationships within the data.

Keywords: Research Paper, Technical Writing, Science, Technology, AI, ML

I. BACKGROUND

The first step towards artificial neural networks came in 1943 when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. They modelled a simple neural network with electrical circuits. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data. The paper's purpose is to introduce neural networks and the principles behind different types of them. Through the real cases of neural networks, this will stimulate readers to think about the importance of neural networks to all aspects of human development.

Perceptron introduced by Frank Rosenblatt in 1957, is the simplest form of ANN used for binary classification. FNNs developed with significant contributions from David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, these networks use backpropagation for training multilayer structures. CNNs popularized by Yann LeCun in the late 1980s excel in image recognition tasks through convolutional and pooling layers. RNNs introduced by John Hopfield in 1982, later enhanced by Jürgen Schmidhuber and Sepp Hochreiter with LSTM networks in 1997 for sequence processing. Radial Basis Function Networks proposed by John Moody and Christian Darken in 1989, employing radial basis functions for effective function approximation. Generative Adversarial Networks introduced by Ian Goodfellow in 2014, leveraging adversarial training between generator and discriminator networks for realistic data generation. The development of various neural network architectures by these pioneering scientists has significantly advanced the capabilities of artificial intelligence.

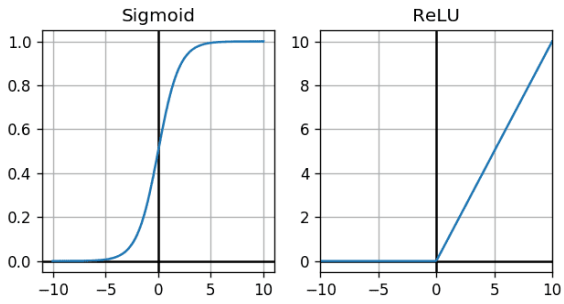
Inside a neural network, activations in one layer of the network determine the activations in the next layer. Considering the input layer takes a specific pattern, hidden layers configure the additional patterns related to the data and provides the conclusion to the output layer. The processes designed inside middle-layers take assigned weights (numbers) and activations from the first layer, between 0 & 1.

$$a_n = \omega_1 a_1 + \omega_2 a_2 + \cdots + \omega_n a_n$$

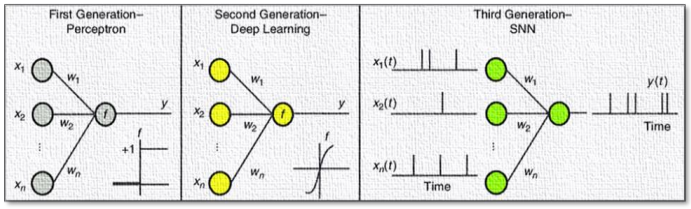
To insert values only between 0 and 1, sigmoid function is used:

$$\sigma_{(x)} = \frac{1}{1 + e^{-x}}$$

Instead of using sigmoid f(x) we now use reLU (rectified linear unit). It is a modern function.
reLU(a) = max (0,a)
f(a) = a



II. CLASSIFICATION OF NEURAL NETWORKS



III. TYPES OF NEURAL NETWORKS

- **Perceptron**:

I. Definition: A perceptron is a binary classifier that maps an input to an output value (given in Figure 1), typically used for linearly separable problems. For specific learning rules, the network can learn with known target values, that is, supervised learning.

II. Mathematical model: Weighted Sum, $z = \sum_{i=1}^{n} \omega_i x_i + b$ and activation function, $y = \begin{cases} 1; z \geq 0 \\ 0; z < 0 \end{cases}$.

III. Variants: Figure 2 explains that Multilayer Perceptron (MLP) extends the perceptron by adding one or more hidden layers, allowing it to solve non-linear problems. Adaline (Adaptive Linear Neuron) uses a linear activation function and updates weights using gradient descent.

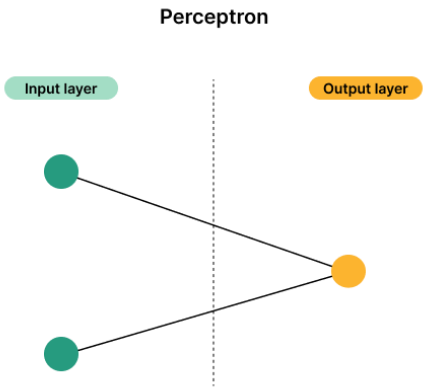


Figure 1

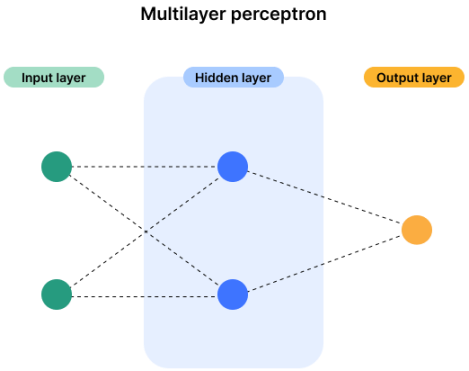


Figure 2

- **Feedforward Neural Networks (FFNN)**:

I. Definition: A feedforward neural network is one of the most basic types of artificial neural networks devised. As shown in Figure 3, in this network, the information moves in only one direction, i.e., forward, from the input nodes, through the hidden nodes and to the output nodes.

II. Training: Often use backpropagation algorithm to adjust weights and minimize error.

III. Weights: Each connection has an associated weight that is adjusted during training.

IV. Application function: Apply non-linear transformations to the weighted sum of inputs.

V. Applications: used in a variety of machine learning tasks including: pattern recognition, classification task, recognition analysis, time series prediction.

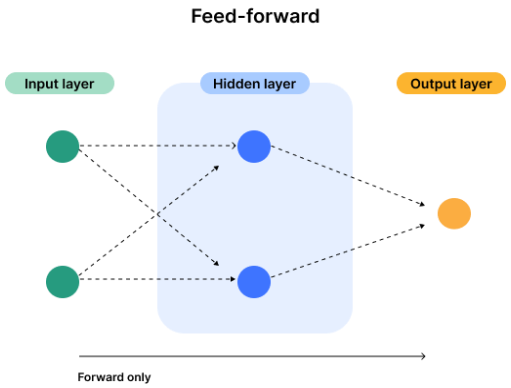


Figure 3

- **Convolutional Neural Networks (CNNs)**:

I. Definition: Specialized type of neural network designed primarily for processing grid-like data, such as images. It achieves parameter sharing and local connectivity. It requires fewer parameters than fully connected networks.
II. Architecture: Consists of convolutional layers, pooling layers, and fully connected layers. Inspired by the organization of the animal visual cortex.
III. Training: Often uses backpropagation and gradient descent and it may require large datasets for optimal performance.
IV. Variants: R-CNN & YOLO v7 for real time object detection, U-net for image segmentation, etc.
V. Applications: Image and video recognition, image classification, object detection, face recognition, edge detection.
VI. Residual: As shown in Figure 4 below, CNNs can be residual networks when they include residual learning mechanisms. Residual networks use skip (or shortcut) connections to allow the gradient to bypass one or more layers, making it easier to train very deep networks.

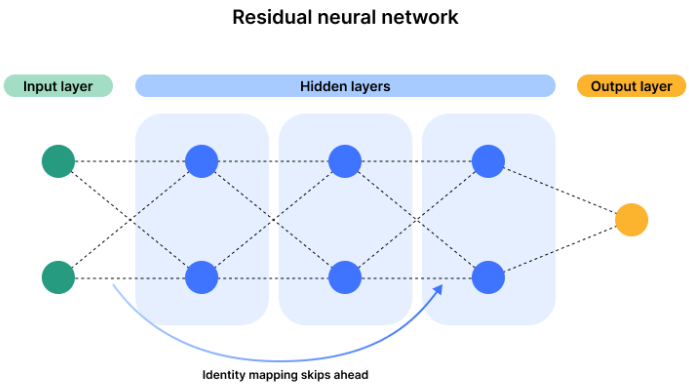


Figure 4

- **Recurrent Neural Networks (RNNs)**:

I. Definition: It is designed to process sequential data and maintain internal state. Consists of loops allowing information to persist across time steps. The design of a recurrent network is given below in Figure 5.
II. Variants: Long Short-Term Memory networks and Gated Recurrent Units which address the vanishing gradient problem and capture long-term dependencies.
III. Advantages: It can handle variable-length sequences, maintain information about previous inputs and share parameters across different time steps.
IV. Training: Uses backpropagation through time (BPTT). It may use techniques like teacher forcing and often apply gradient clipping.
V. Applications: Natural language processing (NLP), speech recognition, time series prediction, machine translation.

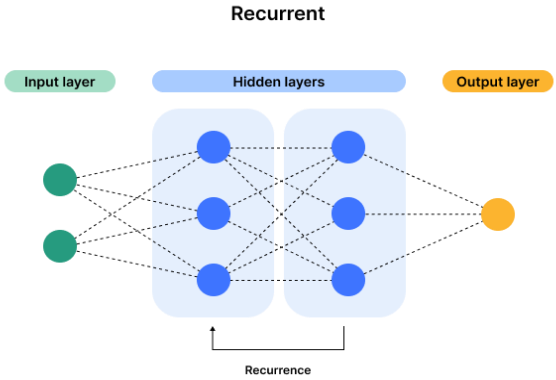


Figure 5

- **Long Short-Term Memory Networks (LSTMs) & Gated Recurrent Unit (GRUs)**:

Both LSTM and GRU networks are powerful tools for handling sequential data, with LSTMs providing a more complex structure for potentially better long-term dependency learning and GRUs offering a simpler, faster alternative.
Comparison of LSTM and GRU:

I. Complexity: LSTMs have more gates and parameters compared to GRUs, which might lead to better performance on complex tasks but require more computational resources.
II. Training Speed: GRUs, with fewer parameters, often train faster than LSTMs.
III. Performance: The performance of LSTM and GRU can be comparable, but the choice depends on the specific task

and dataset. GRUs are often preferred for simpler tasks or when faster training is required.

- **Transformer Neural Network:**

History

Transformer neural networks were introduced by Vaswani et al. in their landmark paper, "Attention is All You Need," published in 2017. The Transformer model revolutionized the field of natural language processing (NLP) by replacing the traditional recurrent neural networks (RNNs) and long short-term memory (LSTM) networks with a novel architecture based entirely on attention mechanisms.

Present

Transformers have become the backbone of most state-of-the-art NLP models today. Some of the most well-known models based on transformers include:

I. BERT (Bidirectional Encoder Representations from Transformers): Developed by Google, BERT set new benchmarks for a variety of NLP tasks.
II. GPT (Generative Pre-trained Transformer): Developed by OpenAI, the GPT series (GPT-2, GPT-3, GPT-4) are known for their impressive text generation capabilities.
III. T5 (Text-To-Text Transfer Transformer): Also developed by Google, T5 treats every NLP problem as a text-to-text problem.
IV. Claude: Developed by Anthropic, majorly using transformer neural networks.

Uses

Transformers have a wide range of applications in various domains:

I. Natural Language Processing (NLP): Transformers are used in tasks such as machine translation, text summarization, question answering, sentiment analysis, and more.
II. Computer Vision: Vision Transformers (ViTs) have been adapted for image classification, object detection, and segmentation tasks.
III. Speech Processing: Transformers are used for tasks like speech recognition and synthesis.
IV. Recommendation Systems: Transformers help in understanding user preferences and making personalized recommendations.
V. Healthcare: Transformers assist in analyzing medical records, predicting disease outcomes, and drug discovery.

Advantages

I. Parallelization: Unlike RNNs and LSTMs, which process data sequentially, transformers process data in parallel, allowing for faster training times.
II. Scalability: Transformers can scale effectively with increased data and computational resources, leading to improved performance on large datasets.
III. Long-Range Dependencies: Transformers can capture long-range dependencies in data due to their attention mechanisms, which is a challenge for traditional RNNs.
IV. Transfer Learning: Pre-trained transformer models can be fine-tuned on specific tasks, making them versatile and efficient for various applications.

How it is better

I. Attention Mechanisms: The attention mechanism allows transformers to weigh the importance of different parts of the input data, leading to better contextual understanding and performance.
II. Flexibility: Transformers can handle both structured and unstructured data, making them applicable to a broad range of tasks beyond NLP.
III. Community and Ecosystem: A large and active research community continuously improves transformer models, leading to rapid advancements and widespread adoption.

IV. NEURAL NETWORKS USED IN MODERN AI SYSTEMS

1. ChatGPT (OpenAI)

ChatGPT, developed by OpenAI, uses the GPT-4 architecture, which is a transformer-based neural network model.

- Architecture: Transformer

- Key Components: Attention mechanisms, self-attention layers, multi-head attention, feedforward neural networks.
- Description: Transformers use self-attention to process input data in parallel, making them highly efficient for language tasks. GPT-4 is fine-tuned for conversational capabilities.

2. META (Facebook AI)
Meta (formerly Facebook) employs various neural networks across its AI products. Notably, the BERT model and its variations are often used for natural language understanding tasks.

- Architecture: Transformer (BERT)
- Key Components: Bidirectional Encoder Representations from Transformers (BERT), attention mechanisms.
- Description: BERT reads text bidirectionally, enabling it to understand context from both directions, which is useful for a wide range of NLP tasks.

3. Gemini (Google DeepMind)
Gemini is likely to use neural networks akin to those developed by DeepMind, such as AlphaFold or other advanced transformer models for various applications.

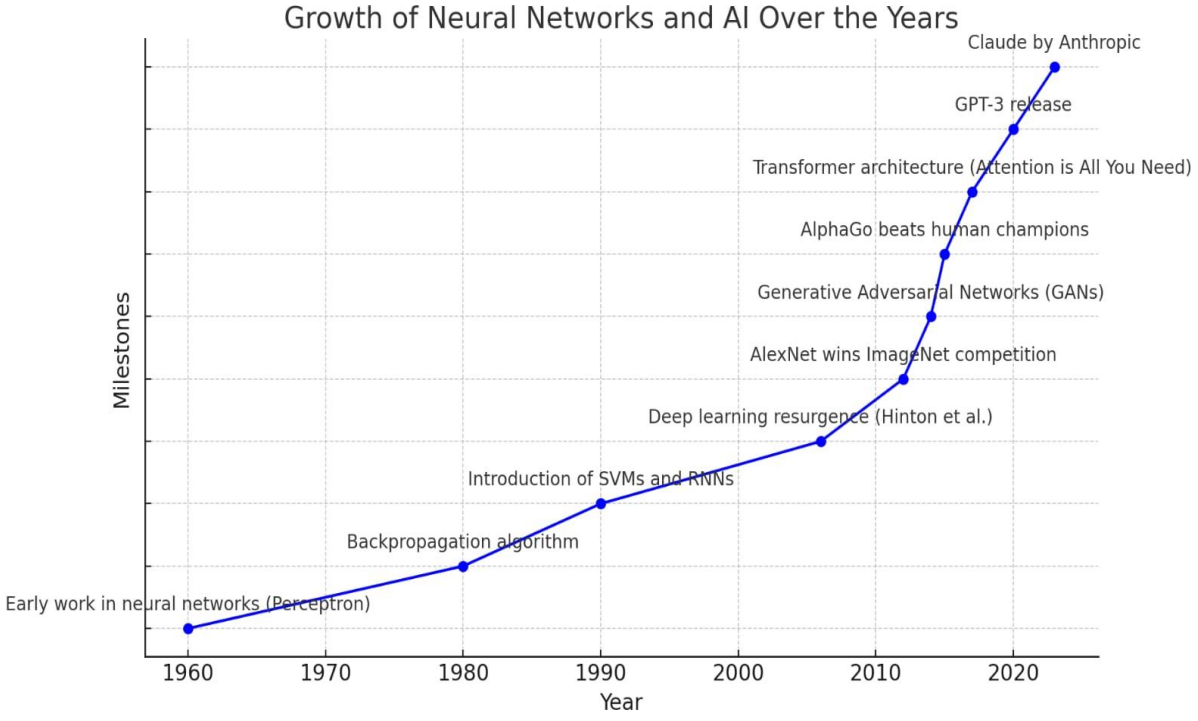
- Architecture: Transformer
- Key Components: Self-attention, multi-head attention, advanced deep learning techniques.
- Description: Similar to GPT and BERT, these models leverage the power of transformers for complex understanding and generation tasks.

4. Claude (Anthropic)
Claude, developed by Anthropic, uses large-scale language models similar to GPT but with specific safety and alignment features.

- Architecture: Transformer
- Key Components: Self-attention mechanisms, alignment techniques to ensure safe and reliable outputs.
- Description: Emphasizes ethical AI development, ensuring models align with human values and provide safe interactions.

5. Copilot (GitHub, OpenAI)
GitHub Copilot, powered by OpenAI, uses Codex, which is a version of GPT-3 fine-tuned for programming.

- Architecture: Transformer (Codex)
- Key Components: Self-attention, fine-tuning on code repositories, integration with development environments.
- Description: Codex is designed to understand and generate code, assisting developers by providing code suggestions and completion.



Graph 1: This visualization shows the chronological progression and key advancements in the field of AI.

V. IMPORTANT STUDIES RELEVANT TO THIS FIELD

Several landmark studies have significantly advanced the field of artificial neural networks (ANNs). Here are some of the most influential ones:

1. The Perceptron (1958) by Frank Rosenblatt:
    - Study: Rosenblatt introduced the perceptron, one of the simplest types of artificial neural networks.
    - Significance: This study laid the groundwork for future neural network research, despite initial limitations in solving non-linear problems.
2. Backpropagation Algorithm (1986) by Rumelhart, Hinton, and Williams:
    - Study: "Learning representations by back-propagating errors."
    - Significance: This paper reintroduced the backpropagation algorithm, enabling multi-layer neural networks to learn complex patterns and features.
3. Convolutional Neural Networks (CNNs) (1998) by Yann LeCun et al.:
    - Study: "Gradient-based learning applied to document recognition."
    - Significance: LeCun's work on CNNs, specifically the LeNet architecture, revolutionized image recognition and processing.
4. Long Short-Term Memory (LSTM) Networks (1997) by Hochreiter and Schmidhuber:
    - Study: "Long short-term memory."
    - Significance: LSTM networks addressed the vanishing gradient problem in training RNNs, making them highly effective for sequence prediction tasks.
5. Deep Belief Networks (2006) by Geoffrey Hinton et al.:
    - Study: "A fast learning algorithm for deep belief nets."
    - Significance: This paper demonstrated how deep neural networks could be pre-trained in an unsupervised manner, sparking renewed interest in deep learning.
6. AlexNet (2012) by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton:
    - Study: "ImageNet classification with deep convolutional neural networks."
    - Significance: AlexNet's success in the ImageNet competition popularized deep CNNs and demonstrated their potential in various applications.
7. Generative Adversarial Networks (GANs) (2014) by Ian Goodfellow et al.:
    - Study: "Generative adversarial nets."
    - Significance: GANs introduced a novel approach to generating synthetic data, greatly impacting fields like image synthesis, data augmentation, and more.
8. ResNet (2015) by Kaiming He et al.:
    - Study: "Deep residual learning for image recognition."
    - Significance: ResNet's architecture solved the degradation problem in deep networks, allowing for the training of extremely deep networks with hundreds of layers.
9. Attention Is All You Need (2017) by Vaswani et al.:
    - Study: Introduced the Transformer architecture.
    - Significance: This paper transformed natural language processing (NLP) and laid the foundation for models like BERT and GPT, enabling significant advancements in language understanding and generation.
10. BERT (2018) by Jacob Devlin et al.:

- Study: "BERT: Pre-training of deep bidirectional transformers for language understanding."
- Significance: BERT introduced bidirectional training of transformers, significantly improving performance on a wide range of NLP tasks.

VI. EXPERIMENT

Objective: The goal of the experiment is to evaluate the performance of a classification model by focusing on the "girl" class as the positive class. We aim to measure the model's performance using various metrics, including accuracy, precision, recall, and F1 score. Additionally, we extend these evaluations to different neural network models such as Feedforward Neural Networks (FFNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN).

1. Confusion Matrix Creation:
A confusion matrix is a fundamental tool used to evaluate the performance of a classification model. It provides a detailed breakdown of the model's predictions compared to the actual labels.

- True Positives (TP): Instances where the model correctly predicts the positive class ("girl").
- False Positives (FP): Instances where the model incorrectly predicts the positive class ("girl") when the true class is "boy".
- True Negatives (TN): Instances where the model correctly predicts the negative class ("boy").
- False Negatives (FN): Instances where the model incorrectly predicts the negative class ("boy") when the true class is "girl".

For the given experiment, the confusion matrix is structured with "girl" as the positive class.

2. Metrics Calculation:
Using the confusion matrix, we calculate the following performance metrics:

- Accuracy: This is the ratio of correctly classified instances (both true positives and true negatives) to the total number of instances.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision: This measures the proportion of positive identifications that were actually correct. It is particularly useful when the cost of false positives is high.

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall (Sensitivity): This measures the proportion of actual positives that were correctly identified. It is crucial when the cost of false negatives is high.

$$\text{Recall} = \frac{\dot{T}P}{TP+FN}$$

F1 Score: This is the harmonic mean of precision and recall, providing a single metric that balances both concerns. It is useful when you need a balance between precision and recall.

$$\text{F1 Score} = 2 \times \frac{P \times R}{P+R}$$

Where, P = precision and R = recall.

3. Application to Neural Network Models:
The same metrics—accuracy, precision, recall, and F1 score—are computed for various types of neural network models:

- Feedforward Neural Networks (FFNN): Metrics for FFNN help evaluate its performance in classification tasks.
- Convolutional Neural Networks (CNN): CNNs are evaluated using the same metrics to understand how well they classify different classes.
- Recurrent Neural Networks (RNN): Metrics are calculated to assess how effectively RNNs handle temporal dependencies in classification tasks.

<u>Steps to Calculate the Confusion Matrix</u>

1. Prepare the Data: Collect a dataset of images with corresponding labels (girl or not girl).
2. Train the Model:
    - Train FFNN, CNN, and RNN models on the dataset.
3. Make Predictions:
    - Use the trained models to make predictions on a test set.
4. Calculate the Confusion Matrix:
    - Compare the predicted labels with the true labels to fill in the confusion matrix.

The code required to calculate the above procedure is given below.

```
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Load your dataset
X, y = np.load('images.npy'), np.load('labels.npy')  # Example dataset

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Data augmentation for training
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow(X_train, y_train, batch_size=32)
test_generator = test_datagen.flow(X_test, y_test, batch_size=32)
```

FFNN MODEL:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense

ffnn_model = Sequential([
    Flatten(input_shape=(image_height, image_width, channels)),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])

ffnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
ffnn_model.fit(train_generator, epochs=10)
```

CNN MODEL:

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D

cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(image_height, image_width, channels)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])

cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
cnn_model.fit(train_generator, epochs=10)
```

MAKE PREDICTION

```
# Predictions for FFNN
ffnn_predictions = ffnn_model.predict(test_generator)
ffnn_predictions = (ffnn_predictions > 0.5).astype(int)

# Predictions for CNN
cnn_predictions = cnn_model.predict(test_generator)
cnn_predictions = (cnn_predictions > 0.5).astype(int)
```

CALCULATION OF CONFUSION MATRIX

```
from sklearn.metrics import confusion_matrix

# True labels
y_true = test_generator.classes

# Confusion matrix for FFNN
```

EXAMPLE OUTPUT OF THIS PROGRAM

	Predicted	
	Girl	Not Girl
Actual Girl	TP	FN
Not Girl	FP	TN

```
Confusion Matrix:
[[4 1]
 [1 4]]
```

From this:

- Accuracy = 0.84+4+1+14+4=0.8 or 80%
- Precision = 0.84+14=0.8 or 80%
- Recall = 0.84+14=0.8 or 80%
- F1 Score = 0.8 or 80%

This is just a simulation. These metrics collectively provide insights into how well the model performs in distinguishing between "girl" and "not girl," indicating strengths and potential areas for improvement.

4. Implementation in Jupyter Notebook:
In a Jupyter notebook, these metrics can be calculated using Python libraries such as scikit-learn. The steps include:

- Importing Libraries: Import necessary functions from scikit-learn to compute the confusion matrix and metrics.
- Defining Labels: Specify true and predicted labels.
- Calculating Metrics: Use accuracy, precision, recall, and f1_score functions to compute the metrics.
- Displaying Results: Present the results in a clear and interpretable format, showing how well the model performs on the classification task.

```python
# Given true and predicted labels
truth = ['girl', 'girl', 'boy', 'girl', 'girl', 'boy', 'boy']
predicted = ['girl', 'boy', 'boy', 'girl', 'girl', 'girl', 'boy']

# Initialize confusion matrix
confusion_matrix = defaultdict(lambda: defaultdict(int))

# Compute the confusion matrix
for t, p in zip(truth, predicted):
    confusion_matrix[t][p] += 1

# Convert confusion matrix to a readable format
def print_confusion_matrix(cm, labels):
    print(f"{'':<10}", end="")
    for label in labels:
        print(f"{label:<10}", end="")
    print()
    for label in labels:
        print(f"{label:<10}", end="")
        for l in labels:
            print(f"{cm[label].get(l, 0):<10}", end="")
        print()

# List of labels
labels = ['girl', 'boy']

# Print the confusion matrix
print_confusion_matrix(confusion_matrix, labels)
```

```
          girl      boy
girl      3         1
boy       1         2
```

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Define the true and predicted labels
truth = ['girl', 'girl', 'boy', 'girl', 'girl', 'boy', 'boy']
predicted = ['girl', 'boy', 'boy', 'girl', 'girl', 'girl', 'boy']
```

```python
# Calculate confusion matrix
cm = confusion_matrix(truth, predicted, labels=['girl', 'boy'])

# Calculate accuracy
accuracy = accuracy_score(truth, predicted)

# Calculate precision, recall, and F1 score for each class
precision = precision_score(truth, predicted, average=None, labels=['girl', 'boy'])
recall = recall_score(truth, predicted, average=None, labels=['girl', 'boy'])
f1 = f1_score(truth, predicted, average=None, labels=['girl', 'boy'])

# Print results
print(f"Confusion Matrix:\n{cm}\n")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Confusion Matrix:
[[3 1]
 [1 2]]

Accuracy: 0.71
Precision: [0.75       0.66666667]
Recall: [0.75       0.66666667]
F1 Score: [0.75       0.66666667]
```

VII. CONCLUSION

In this paper, we have traversed the expansive and rapidly evolving landscape of machine learning and neural networks. The exploration of neural networks in artificial intelligence reveals the profound impact these technologies have had on the evolution and capabilities of AI systems. Beginning with the foundational understanding provided by the abstract and background, we gain insight into the design and historical development of neural networks, including key scientists and significant dates that have shaped the field. The discussion encompasses crucial activation functions like ReLU and sigmoid, which are instrumental in the functioning of neural networks.

The classification of neural networks, tracing their generational advancements from perceptrons to deep learning models and spiking neural networks (SNN), sets the stage for a detailed examination of various types of neural networks. The study delves deeply into feed-forward neural networks (FFNN), convolutional neural networks (CNN), recurrent neural networks (RNN), long short-term memory networks (LSTM), gated recurrent units (GRU), and the increasingly prevalent transformer neural networks.

AI bots, such as ChatGPT, Claude, and Gemini, demonstrate practical applications of these neural networks, underscoring their significance in modern AI implementations. The graphical representation of AI tools' growth over the years since the 1960s provides a visual context for understanding the expanding influence of neural networks. The review of significant studies and landmark research papers from 1958 to the 2000s highlights the milestones and intellectual contributions that have propelled the field forward.

The experimentation phase, focusing on accuracy, precision, recall, F1 score performance, and confusion matrix predictions, offers empirical evidence of neural networks' efficacy. By evaluating true positives, true negatives, false positives, and false negatives, the study validates the performance of FFNN and CNN models in image classification tasks, such as determining whether an image depicts a girl or not.

Overall, this research underscores the transformative role of neural networks in artificial intelligence, from their historical roots to their modern-day applications and future potential. The comprehensive analysis affirms that neural networks are not only fundamental to current AI advancements but also pivotal for future innovations in the field.

REFERENCES

[1] Lippmann, R.P., 1987. An introduction to computing with neural nets. IEEE Accost. Speech Signal Process. Mag., April: 4-22.

[2] Ms. Sonali. B. Maind, Ms. Priyanka Wanka, Basic of Artificial Neural Network, International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169.

[3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

[4] Yumin Pan, Different Types of Neural Networks and Applications: Evidence from Feedforward, Convolutional and Recurrent Neural Networks, March 2024, Highlights in Science Engineering and Technology 85:247-255

[5] Vaswani et al author, 2017, Attention is all you Need.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

[7] Sd Sourav, Accuracy Matrices, Repository, May: 5-2020.

[8] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[9] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).